STORING, TRACKING, AND QUERYING PROVENANCE IN LINKED DATA

M. Sreerama Murty
Research Scholar, Department of CSE,
Achary Nagarjuna University, Guntur
sreeramsssit@gmail.com

Dr. N. Nagamalleswara Rao
Professor, Department of CSE, RVR & JC
College of Engineering, Guntur
nnmrao@rvrjc.ac.in

Abstract
Different notions of provenance for database queries have been proposed and studied in the past few years. In this article, we detail three main notions of database provenance, some of their applications, and compare and contrast amongst them. Specifically, we review why, how, and where provenance, describe the relationships among these notions of provenance, and describe some of their applications in confidence computation, view maintenance and update, debugging, and annotation propagation. *Provenance in Databases* reviews research over the past ten years on why, how, and where provenance, clarifies the relationships among these notions of provenance, and describes some of their applications in confidence computation, view maintenance and update, debugging, and annotation propagation. Provenance in Databases is intended for engineers and researchers who would like to familiarize themselves with the foundations, as well as the many challenges in the field of database provenance. In particular, the capacity to store, track, and query provenance data is becoming a pivotal feature of modern triple stores. We present methods extending a native RDF store to efficiently handle the storage, tracking, and querying of provenance in RDF data. We describe a reliable and understandable specification of the way results were derived from the data and how particular pieces of data were combined to answer a query. Subsequently, we present techniques to tailor queries with provenance data.

***Keywords**: Provenance Query, RDF Store, Tailor Query, Database*

Introduction

1 Provenance query:

The concept of a provenance query was defined by Simon Miles in order to only select a relevant subset of all possible results when looking up the provenance of an entity [37]. A number of authors have presented systems for specifically handling such provenance queries. Biton et al. showed how user views can be used to reduce the amount of information returned by provenance queries in a workflow system [38]. The MTCProv [39] and the RDFProv [40] systems focus on managing and enabling queries over provenance that result from scientific workflows. Similarly, the ProQL approach [41] defines a query language and proposes relational indexing techniques for speeding up provenance queries involving path traversals.

2 Use Cases for Provenance Polynomials

There are many scenarios provenance polynomials can be applied to. [18] describes a number of use cases where storage and querying of provenance data generated by a database system could be useful. We revisit some of these here. Polynomials express the exact way through which the results were derived. As such, they can hence be used to calculate scores or probabilities for particular query results (e.g., for post-processing tasks such as results ranking or faceted search). Likewise, one can use polynomials to compute a trust or information quality score based on the sources used in the result. One can also use the provenance to modify query execution strategies on the fly. For instance, one could restrict the results to certain subsets of sources or use provenance for access control such that only certain sources will appear in a query result. Identifying results (i.e., particular triples) with overlapping provenance is also another prospective use case. Finally, one could detect whether a particular result would still be valid when removing a source dataset.

3 Provenance Polynomials

The first question we tackle is how to represent provenance information that we want to return to the user in addition to the results themselves. Beyond listing the various sources involved in the query, we want to be able to characterize the specific ways in which each source contributed to the query results. As summarized in Section 2, there has been quite a bit of work on provenance models.and languages recently. Here, we leverage the notion of provenance polynomials. However, in contrast to the many recent pieces of work in this space, which tackled more theoretical issues, we focus on the practical realization of this model within a high performance triple store to answer queries seen as useful in practice. Specifically,

We focus on two key requirements:

1. The capability to pinpoint, for each query result, the exact source from which the result was selected;
2. The capability to trace back, for each query result, the complete list of sources and how they were combined to deliver a result.

4 Storage Models

We now discuss the RDF storage model based on our previous contribution [28], and extended with new physical storage structures to store provenance.

Native Storage Model

RDF Templates when ingesting new triples, first identifies RDF sub graphs. It analyzes the incoming data and builds what are termed molecule templates. These templates act as data prototypes to create RDF molecules. Figures 2 i) gives a template example that co-locates information relating to Student instances. Once the templates have been defined, the system starts creating molecule identifiers based on the molecule roots (i.e., central molecule nodes) that it identifies in the incoming data

Data (or workload) inspection algorithms can be exploited in order to materialize frequent joins though molecules. In addition to materializing the joins between an entity and its corresponding values (e.g., between a student and his/her firstname), one can hence materialize the joins between two semantically related entities (e.g., between a student and his/her advisor) that are frequently co-accessed by co-locating them in the same molecule.

Storage Model

Variants for Provenance We now turn to the problem of extending the physical data structures of proposed system to support provenance queries. There are a number of ways one could implement this in our system. A first way of storing provenance data would be to simply annotate every object in the database with its corresponding source. This produces quadruple physical data structures (SP OL, where S is the subject of the quadruple, P its predicate, O its object, and L its source), as illustrated in Figure 3, SP OL). The main advantage of this variant is its ease of implementation (e.g., one simply has to extend the data structure storing the object to also store the source data). Its main disadvantage, however, is memory consumption since the source data has to be repeated for each triple.

5. Query Execution

We now turn to the way we take advantage of the source information stored in the molecules to produce provenance polynomials. We have implemented specific query execution strategies in our system that allow to return a complete record of how the results were produced (including detailed information of key operations like unions and joins) in addition to the results themselves. The provenance polynomials our system produce can be generated at source-level or at triple-level, and both for detailed provenance records and for aggregated provenance records.

The first example query we consider is a simple star query, i.e., a query defining a series of triple patterns, all joined on an entity that has to be identified:

```
select ?lat ?long
where {
 ?a [] ''Eiffel Tower''. (<- 1st constraint)
 ?a inCountry FR .     (<- 2nd constraint)
 ?a lat ?lat .        (<- 1st projection)
 ?a long ?long .    (<- 2nd projection)
}
```

To build the corresponding provenance polynomial, first identifies the constraints and projections from the query (see the annotated listing above). The query executor chooses the most selective pattern to start looking up molecules (in this case the first pattern), translates the bound variable ("Eiffel Tower") into a key, and retrieves all molecules containing that key. Each molecule is then inspected in turn to determine whenever both

i) The various constraints can be met (checkIfTripleExists in the algorithm) and
ii) The projections can be correctly processed (get Entity in the algorithm). Our system keeps track of the provenance of each result, by joining the local provenance information of each triple used during query execution to identify the result.

6 Performance Evaluations

To empirically evaluate our approach, we implemented the storage models and query execution strategies described above. Specifically, we implemented two different storage models: SPOL and SLOP. For each model, we support two different levels of provenance granularity: source granularity and triple granularity. Our system does not parse SPARQL queries at this stage (adapting a SPARQL parser is currently in progress), but offers a similar, high-level and declarative API to encode queries using triple patterns. Each query is then encoded into a logical physical plan (a tree of operators), which is then optimized into a physical query plan as for any standard database system.

7 Conclusions

To the best of our knowledge, we presented the first attempt to translate theoretical insight from the database provenance literature into a high-performance triplestore. Our techniques enable not only simple tracing of lineage for query results, but also considers fine-grained multilevel provenance and allow us to tailor the query execution with provenance information. We introduced two storage models and five query execution strategies for supporting provenance in Linked Data management systems. From our experiment we can say that the more data we have to process the slower queries are executed and the more selective is the provenance query the more we gain in performance. Less selective workload queries are more sensitive to those aspects than queries that allow to early prune intermediate results.

The more advanced query execution strategies that take advantage of the selectivity of the provenance information are more sensitive to the number elements returned by the provenance query. It is important to remember that the performance of a system processing provenance, either by tracking or querying such meta data, highly depends on the selectivity of the provenance [12], [13]. On the one hand, the more selective provenance information is the more one can gain by scoping the query execution with provenance-enabled queries. On the other hand, with more selective data, tracking and storing provenance information becomes more expensive. Therefore such systems should be evaluated with respect to the data sampled from the targeted source to choose the most suitable storage model and a query execution strategy. A user of a system like TripleProv can easily restrict what data to be used in a query by providing a SPARQL query scoping the data provenance (Section 5). Additionally, the user will receive detailed information about exact pieces of data were used to produce the results (Section 4). Such a provenance trace can be used tocompute the quality of the results or to (partially) invalidate the results in case some parts of the data collection turn out to be incorrect.

Moreover, a provenance trace can be leveraged to partially re-evaluate the query in case new data arrives, i.e., we can re-evaluate only the part that is influenced by the new data since we know what data exactly has been used in the query. Building on the results of this work, we see a number of possible avenues for future work: Investigating how provenance can be used to improve query execution and optimization within a data management system is one of them. Another possibility is to leverage provenance to improve the reconstruction of incomplete graph data, especially in the context of streams of Linked Data. Finally, we plan to investigate the ability to track and query provenance in dynamic Linked Data environments, linking provenance management to complex event processing and reasoning.

8 References

[1] J. J. Carroll, C. Bizer, P. Hayes, and P. Stickler. Named graphs, provenance and trust. In Proceedings of the 14th international conference on World Wide Web, pages 613–622. ACM, 2005.

[2] J. Cheney, L. Chiticariu, and W.-C. Tan. Provenance in databases: Why, how, and where, volume 1. Now Publishers Inc, 2009.

[3] P. Cudr´e-Mauroux, K. Lim, R. Simakov, E. Soroush, P. Velikhov, D. L. Wang, M. Balazinska, J. Becla, D. DeWitt, B. Heath, D. Maier, S. Madden, J. M. Patel, M. Stonebraker, and S. Zdonik. A Demonstration of SciDB: A Science-Oriented DBMS. Proceedings of the VLDB Endowment (PVLDB), 2(2):1534–1537, 2009.

[4] C. V. Dam´asio, A. Analyti, and G. Antoniou. Provenance for sparql queries. In Proceedings of the 11th international conference on The Semantic Web - Volume Part I, ISWC'12, pages 625–640, Berlin, Heidelberg, 2012. Springer-Verlag.

[5] G. Demartini, I. Enchev, M. Wylot, J. Gapany, and P. Cudre-Mauroux. Bowlognabenchˆ aAˇTbenchmarking ˇ rdf analytics. In K. Aberer, E. Damiani, and T. Dillon, editors, Data-Driven Process Discovery and Analysis, volume 116 of Lecture Notes in Business Information Processing, pages 82–102. Springer Berlin Heidelberg, 2012.

[6] L. Ding, Y. Peng, P. P. da Silva, and D. L. McGuinness. Tracking RDF Graph Provenance using RDF Molecules. In International Semantic Web Conference, 2005.

[7] G. Flouris, I. Fundulaki, P. Pediaditis, Y. Theoharis, and V. Christophides. Coloring rdf triples to capture provenance. In Proceedings of the 8th International Semantic Web Conference, ISWC '09, pages 196–212, Berlin, Heidelberg, 2009. Springer-Verlag.

[8] F. Geerts, G. Karvounarakis, V. Christophides, and I. Fundulaki. Algebraic structures for capturing the provenance of sparql queries. In Proceedings of the 16th International Conference on Database Theory, ICDT '13, pages 153–164, New York, NY, USA, 2013. ACM.

[9] T. J. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pages 31– 40. ACM, 2007.

[10] P. Groth, Y. Gil, J. Cheney, and S. Miles. Requirements for provenance on the web. International Journal of Digital Curation, 7(1), 2012.

[11] P. Groth and L. Moreau (eds.). PROV-Overview. An Overview of the PROV Family of Documents. W3C Working Group Note NOTE-prov-overview-20130430, World Wide Web Consortium, Apr. 2013.

[12] P. T. Groth. Transparency and reliability in the data supply chain. IEEE Internet Computing, 17(2):69–71, 2013.

[13] O. Hartig. Provenance information in the web of data. In Proceedings of the 2nd Workshop on Linked Data on the Web (LDOW2009), 2009.

[14] O. Hartig. Querying trust in rdf data with tsparql. In Proceedings of the 6th European Semantic Web Conference on The Semantic Web: Research and Applications, ESWC 2009 Heraklion, pages 5– 20, Berlin, Heidelberg, 2009. Springer-Verlag.

[15] P. Hayes and B. McBride. Rdf semantics. W3C Recommendation, February 2004

About the Authors

M.Sreerama Murthy pursuing Ph.D in Acharya Nagarjuna University,Guntur.M.Tech in Computer Scince and Engineering from University College of Engineering ,JNTU, Kakinada. B.Tech in Information Technology from JNTU, Hyderabad. His research areas includes Data Mining and Big Data

Dr. N. Naga MalleswaraRao , working as Professor in Department of CSE, RVR & JC College of Engineering Chowdavaram, Guntur(Dt),Andhra Pradesh, India. He was 27 years of teaching experience and published few national and international journals and also attended national and international conferences. His research areas includes computer algorithms, compilers, image processing and data mining